An Introduction to APIs

By Brian Cooksey

zapier

Chapter 1: Introduction

API (application programming interface) adalah sebuah bagian besar dari web. Pada 2013 ada lebih dari 10.000 API yang diterbitkan oleh perusahaan untuk konsumsi secara bebas, Dengan begitu banyak perusahaan yang berinvestasi di daerah bisnis baru ini, memiliki pengetahuan pemahaman cara kerja API menjadi semakin relevan dengan karir di industri perangkat lunak. Melalui kursus ini, kami berharap dapat memberikan pengetahuan bahwa dengan membangun pengetahuan dari yang sangat dasar. Dalam bab ini, kita mulai dengan melihat beberapa konsep dasar sekitar API. Kita mendefinisikan apa API, di mana ia tinggal, dan memberikan gambaran tingkat tinggi tentang bagaimana satu API digunakan.

A Frame Of reference

Ketika berbicara tentang API, banyak pembahasan berfokus pada konsep-konsep abstrak. Untuk jangkar diri kita sendiri, mari kita mulai dengan sesuatu yang fisik: server. Sebuah server tidak lebih dari sebuah komputer besar. Ia memiliki semua bagian yang sama dengan laptop atau desktop yang Anda gunakan untuk bekerja, itu hanya lebih cepat dan lebih kuat. Biasanya, server tidak memiliki monitor, keyboard, atau mouse, yang membuat mereka terlihat didekati. Kenyataannya adalah bahwa orang-orang IT terhubung ke mereka dari jarak jauh - jauh berpikir desktop-gaya - untuk bekerja pada mereka.

Server yang digunakan untuk segala macam hal. Beberapa data store; lain mengirim email. Jenis orang berinteraksi dengan sebagian besar server web. Ini adalah server yang memberikan Anda sebuah halaman web ketika Anda mengunjungi sebuah website. Untuk lebih memahami bagaimana cara kerjanya, di sini adalah analogi sederhana

Dengan cara yang sama bahwa program seperti Solitaire menunggu Anda untuk mengklik pada kartu untuk melakukan sesuatu, web server menjalankan program yang menunggu seseorang untuk meminta itu untuk halaman web.

Ada benar-benar ajaib atau spektakuler tentang hal itu. Seorang pengembang perangkat lunak menulis program, salinan ke server, dan server menjalankan program secara terus-menerus.

Apa API Apakah dan Mengapa Ini Berharga

Website yang dirancang untuk memenuhi kekuatan rakyat. Manusia memiliki kemampuan luar biasa untuk mengambil informasi visual, menggabungkan dengan pengalaman kami untuk mendapatkan makna, dan kemudian bertindak atas makna itu. Itu sebabnya Anda dapat melihat formulir di situs web dan tahu bahwa kotak kecil dengan kalimat "First Name" di atas itu berarti Anda diharapkan untuk mengetikkan kata yang Anda gunakan untuk informal mengidentifikasi diri.

Namun, apa yang terjadi ketika Anda menghadapi waktu-intensif tugas yang sangat, seperti menyalin info kontak untuk seribu pelanggan dari satu situs ke situs lain? Anda akan senang untuk mendelegasikan pekerjaan ini ke komputer sehingga dapat dilakukan dengan cepat dan akurat. Sayangnya, karakteristik yang membuat website yang optimal bagi manusia membuat mereka sulit untuk komputer untuk digunakan.

Solusinya adalah API. API adalah alat yang membuat data website dicerna untuk komputer. Melalui itu, komputer dapat melihat dan mengedit data, seperti seseorang bisa dengan memuat halaman dan mengirimkan formulir.

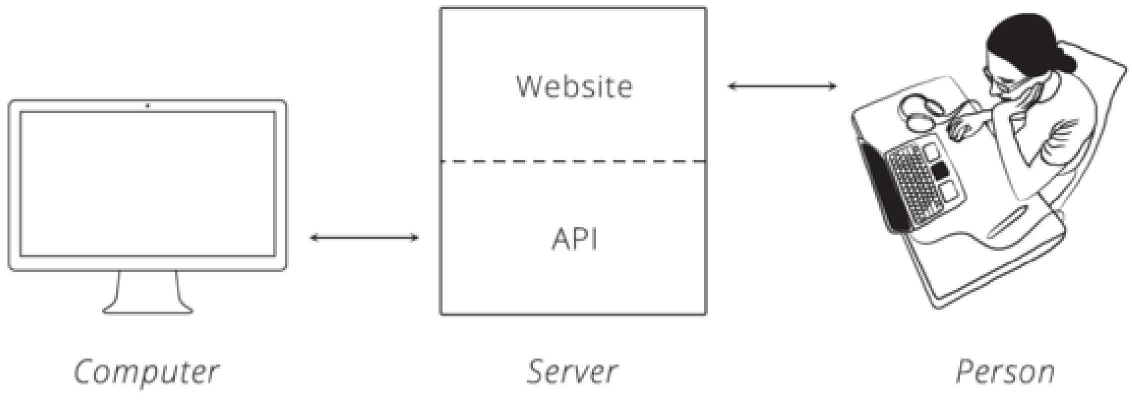


Figure 1. Communicating with a server.

Membuat data lebih mudah untuk bekerja dengan baik karena itu berarti orang dapat menulis perangkat lunak untuk mengotomatisasi tugas-tugas membosankan dan padat karya. Apa yang mungkin mengambil jam manusia untuk mencapai dapat mengambil detik komputer melalui API.

Bagaimana Sebuah API Digunakan

Ketika dua sistem (website, desktop, smartphone) menghubungkan melalui API, kami mengatakan mereka "terintegrasi." Dalam sebuah integrasi, Anda memiliki dua sisi, masing-masing dengan nama khusus. Satu sisi kita telah berbicara tentang: server. Ini adalah sisi yang benar-benar memberikan API. Ini membantu untuk mengingat bahwa API ini hanya program lain yang berjalan pada server 3. Ini mungkin bagian dari program yang sama yang menangani lalu lintas web, atau dapat menjadi salah satu benar-benar terpisah. Dalam kedua kasus, itu duduk, menunggu orang lain untuk meminta untuk data.

Sisi lain adalah "klien." Ini adalah program terpisah yang tahu apa data tersedia melalui API dan dapat memanipulasi, biasanya atas permintaan dari pengguna. Sebuah contoh yang bagus adalah aplikasi smartphone yang sync dengan website. Ketika Anda menekan tombol refresh aplikasi Anda, itu berbicara ke server melalui API dan mengambil info terbaru.

Prinsip yang sama berlaku untuk situs web yang terintegrasi. Ketika salah satu situs menarik di data dari yang lain, situs yang menyediakan data bertindak sebagai server, dan situs mengambil data adalah klien.

Kesimpulan BAB 1

Bab ini berfokus pada penyediaan beberapa terminologi dasar dan model mental dari apa yang API dan bagaimana ia digunakan.

Istilah kunci yang kita pelajari adalah:

- Server: Sebuah komputer yang kuat yang berjalan API
- API: The "tersembunyi" dari sebuah situs web yang dimaksudkan untuk konsumsi computer
- Klien: Sebuah program yang pertukaran data dengan server melalui API

Bab 2: Protokol

Bagaimana komputer berbicara satu sama lain

Dalam Bab 1, kami mendapat bantalan kami dengan membentuk gambar dari kedua belah pihak yang terlibat dalam API, server dan klien. Dengan pemahaman yang solid pada siapa, kami siap untuk melihat lebih dalam bagaimana kedua berkomunikasi. Untuk konteks, pertama-tama kita melihat model manusia komunikasi dan membandingkannya dengan komputer. Setelah itu, kami pindah ke spesifik dari protokol yang umum digunakan dalam API.

Mengetahui Aturan

Orang membuat etiket sosial untuk membimbing interaksi mereka. Salah satu contoh adalah bagaimana kita berbicara satu sama lain di telepon. Bayangkan diri Anda chatting dengan teman. Sementara mereka berbicara, Anda tahu untuk diam. Anda tahu untuk memungkinkan mereka jeda singkat. Jika mereka mengajukan pertanyaan dan kemudian tetap tenang, Anda tahu mereka mengharapkan respon dan sekarang giliran Anda untuk berbicara.

Komputer memiliki etiket yang sama, meskipun ia pergi dengan istilah "protokol." Sebuah protokol komputer adalah seperangkat diterima aturan yang mengatur bagaimana dua komputer dapat berbicara satu sama lain. Dibandingkan dengan standar kami, bagaimanapun, protokol komputer sangat kaku. Pikirkan sejenak dari dua kalimat "Warna favorit saya adalah biru" dan "Biru adalah warna favorit saya." Orang-orang dapat memecah setiap kalimat dan melihat bahwa mereka berarti hal yang sama, meskipun kata-kata yang dalam urutan yang berbeda. Sayangnya, komputer tidak begitu pintar.

Untuk dua komputer untuk berkomunikasi secara efektif, server harus tahu persis bagaimana klien akan mengatur pesan-nya. Anda dapat menganggap itu seperti orang meminta alamat surat. Ketika Anda meminta lokasi tempat, Anda menganggap hal pertama yang Anda diberitahu adalah alamat jalan, diikuti oleh kota, negara, dan terakhir, kode pos. Anda juga memiliki harapan tertentu tentang masing-masing bagian dari alamat, seperti fakta bahwa kode pos hanya harus terdiri dari angka. Sebuah tingkat yang sama spesifisitas diperlukan untuk protokol komputer untuk bekerja

Protokol dari Web

Ada protokol untuk hanya tentang segala sesuatu; masing-masing disesuaikan untuk melakukan pekerjaan yang berbeda. Anda mungkin sudah mendengar dari beberapa: Bluetooth untuk menghubungkan perangkat, dan POP atau IMAP untuk mengambil email.

Di web, protokol utama adalah Transfer Protocol Hyper-Text, lebih dikenal dengan singkatan, HTTP. Ketika Anda mengetik alamat seperti http://example.com ke web browser, "http" memberitahu browser untuk menggunakan aturan HTTP ketika berbicara dengan server.

Dengan mana-mana HTTP di web, banyak perusahaan memilih untuk mengadopsi sebagai protokol yang mendasari API mereka. Salah satu manfaat dari menggunakan protokol familiar adalah bahwa hal itu menurunkan kurva pembelajaran bagi pengembang, yang mendorong penggunaan API. Manfaat lain adalah bahwa HTTP memiliki beberapa fitur yang berguna dalam membangun API yang baik, seperti yang akan kita lihat nanti. Sekarang, mari kita berani air dan lihatlah bagaimana HTTP bekerja!

Permintaan HTTP

Komunikasi di pusat-pusat HTTP sekitar konsep yang disebut Siklus Permintaan-Response. Client mengirimkan server permintaan untuk melakukan sesuatu. Server, pada gilirannya, mengirimkan klien respon mengatakan apakah atau tidak server bisa melakukan apa yang klien minta.

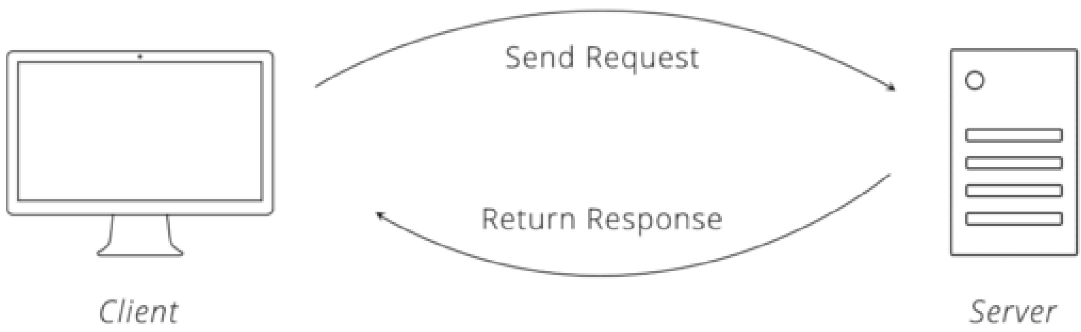


Figure 1. The Request-Response Cycle.

Untuk membuat permintaan yang valid, klien perlu menyertakan empat hal:

1. URL (Uniform Resource Locator) 1
2. Metode
3. Daftar Header
4. Tubuh

Itu mungkin terdengar seperti banyak rincian hanya untuk menyampaikan pesan, tapi ingat, komputer harus sangat spesifik untuk berkomunikasi dengan satu sama lain

URL

URL yang akrab bagi kita melalui penggunaan sehari-hari kami web, tetapi apakah Anda pernah mengambil waktu sejenak untuk mempertimbangkan struktur mereka? Dalam HTTP, URL adalah alamat unik untuk hal (kata benda). Yang hal mendapatkan alamat sepenuhnya terserah bisnis berjalan server. Mereka dapat membuat URL untuk halaman web, gambar, atau bahkan video binatang lucu.

API memperluas ide ini sedikit lebih jauh untuk memasukkan benda seperti pelanggan, produk, dan tweets. Dengan demikian, URL menjadi cara mudah bagi klien untuk memberitahu server yang hal itu ingin berinteraksi dengan. Tentu saja, API juga tidak menyebut mereka "hal-hal", tetapi memberi mereka nama teknis "sumber.

Metode

Metode permintaan memberitahu server apa tindakan klien ingin server untuk mengambil. Bahkan, metode ini sering disebut sebagai permintaan "kata kerja."

Empat metode yang paling sering terlihat di API adalah:

- GET - Minta server untuk mengambil sumber daya
- POST - Minta server untuk menciptakan sumber daya baru
- PUT - Minta server untuk mengedit / memperbarui sumber daya yang ada
- DELETE - Minta server untuk menghapus sumber daya

Berikut ini adalah contoh untuk membantu menggambarkan metode ini. Katakanlah ada sebuah restoran pizza dengan API yang dapat digunakan untuk menempatkan pesanan. Anda memesan dengan membuat permintaan POST ke server restoran dengan detail pesanan Anda, meminta mereka untuk membuat pizza Anda. Segera setelah Anda mengirimkan permintaan, namun, Anda sadar bahwa Anda memilih salah gaya kerak, sehingga Anda membuat permintaan PUT untuk mengubahnya.

header

Header menyediakan meta-informasi tentang permintaan. Mereka adalah daftar sederhana item seperti saat klien mengirimkan permintaan dan ukuran tubuh permintaan.

Apakah Anda pernah mengunjungi sebuah website di smartphone Anda yang diformat khusus untuk perangkat mobile? Yang dimungkinkan oleh header HTTP yang disebut "User-Agent." Klien menggunakan header ini untuk memberitahu server apa jenis perangkat yang Anda gunakan, dan situs cukup pintar untuk mendeteksi dapat mengirimkan format terbaik untuk perangkat Anda.

Ada cukup header HTTP beberapa klien dan server menangani, jadi kami akan menunggu untuk membicarakan yang lain sampai mereka relevan di bab berikutnya.

Sementara menunggu pesanan Anda, Anda membuat sekelompok permintaan GET untuk memeriksa status. Setelah satu jam menunggu, Anda memutuskan Anda sudah cukup dan membuat permintaan DELETE untuk membatalkan pesanan Anda.

Tubuh

Permintaan tubuh berisi data klien ingin mengirim server. Melanjutkan kami contoh pizza pemesanan di atas, tubuh mana rincian pesanan pergi.

Sebuah sifat unik tentang tubuh adalah bahwa klien memiliki kontrol penuh atas ini bagian dari permintaan. Tidak seperti metode, URL, atau header, di mana protokol HTTP memerlukan struktur kaku, tubuh memungkinkan klien untuk mengirim sesuatu yang dibutuhkan.

Keempat buah - URL, metode, header, dan tubuh - membuat permintaan HTTP lengkap

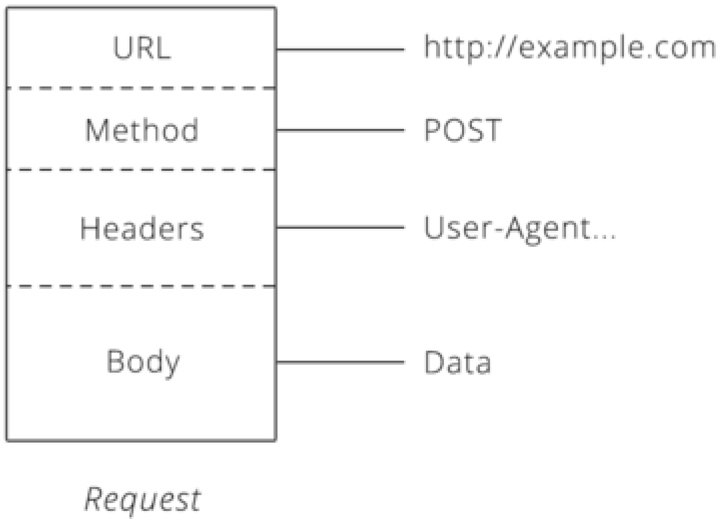


Figure 2. The structure of an HTTP request.

Tanggapan HTTP

Setelah server menerima permintaan dari klien, ia mencoba untuk memenuhi permintaan dan mengirim respon klien kembali. Tanggapan HTTP memiliki struktur yang sangat mirip dengan permintaan. Perbedaan utama adalah bahwa alih-alih metode dan URL, respon termasuk kode status. Di luar itu, header respon dan tubuh mengikuti format yang sama seperti permintaan.

Status Codes

Kode status yang tiga digit angka yang masing-masing memiliki makna yang unik. Ketika digunakan dengan benar dalam API, jumlah ini sedikit dapat berkomunikasi banyak info ke klien. Sebagai contoh, Anda mungkin telah melihat halaman ini selama pengembaraan internet Anda:

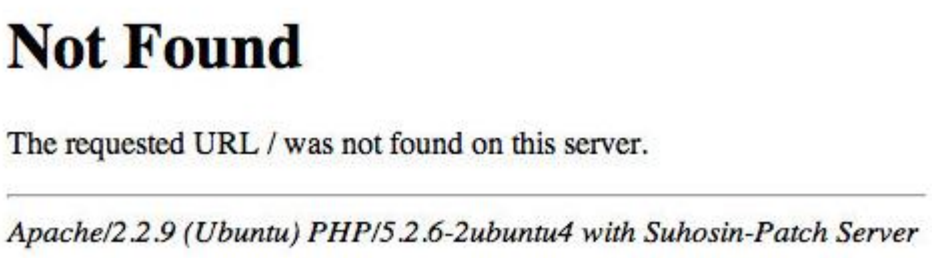


Figure 3. A default 404 web page.

Kode status respon balik ini adalah 404, yang berarti "Tidak Ditemukan." Setiap kali klien membuat permintaan untuk sumber daya yang tidak ada, server merespon dengan kode status 404 untuk membiarkan klien tahu: "sumber daya yang tidak ada, jadi tolong jangan meminta lagi!"

Ada membunuh status lainnya dalam protokol HTTP, termasuk 200 ("sukses! Permintaan yang baik") ke 503 ("website / API saat ini turun.") Kita akan belajar beberapa dari mereka saat mereka datang di bab-bab selanjutnya.

Setelah respon yang dikirim ke klien, Siklus Permintaan-Response selesai dan bahwa putaran komunikasi lebih. Sekarang sampai klien untuk memulai interaksi lebih lanjut. Server tidak akan mengirim klien data lagi sampai menerima permintaan baru

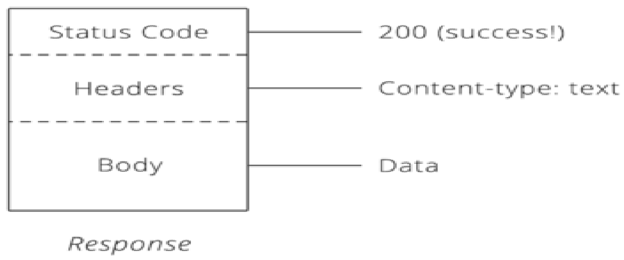


Figure 4. The structure of an HTTP response.

Bagaimana API Membangun HTTP

Sekarang, Anda dapat melihat HTTP yang mendukung berbagai permutasi untuk membantu klien dan server bicara. Jadi, bagaimana ini membantu kami dengan API? Fleksibilitas HTTP berarti bahwa API dibangun di atasnya dapat menyediakan klien dengan banyak potensi bisnis. Kami melihat bahwa potensi pizza memesan contoh di atas. Sebuah Tweak sederhana untuk metode permintaan adalah perbedaan antara mengatakan server untuk menciptakan sebuah tatanan baru atau membatalkan satu yang ada. Itu mudah untuk mengubah hasil bisnis yang diinginkan ke dalam instruksi server bisa mengerti. Sangat kuat!

Fleksibilitas ini dalam protokol HTTP meluas ke bagian lain dari permintaan, juga. Beberapa API memerlukan header tertentu, sementara yang lainnya memerlukan informasi spesifik di dalam tubuh permintaan. Mampu menggunakan API bergantung pada mengetahui bagaimana membuat permintaan HTTP yang benar untuk mendapatkan hasil yang Anda inginkan.

Kesimpulan BAB 2

Tujuan dari bab ini adalah untuk memberikan pemahaman dasar HTTP. Konsep utama adalah Siklus Permintaan-Response, yang kita pecah ke dalam bagian-bagian berikut:

- Permintaan - terdiri dari URL (http: // ...), metode (GET, POST, PUT, DELETE), daftar header (User-Agent ...), dan tubuh (data).
- Respon - terdiri dari kode status (200, 404 ...), daftar header, dan tubuh.

Sepanjang sisa saja, kami akan kembali fundamental ini karena kami menemukan bagaimana API mengandalkan mereka untuk memberikan kekuatan dan fleksibilitas.

Bab (chafter) 3: Data Format

Sejauh ini, kita telah belajar bahwa HTTP (Hyper-Text Transfer Protocol) merupakan fondasi dari API di web dan yang menggunakannya, kita perlu tahu bagaimana HTTP bekerja. Dalam bab ini, kita mengeksplorasi API data memberikan, bagaimana itu diformat, dan bagaimana HTTP memungkinkan

mewakili data

Ketika berbagi data dengan orang, kemungkinan bagaimana menampilkan informasi tersebut hanya dibatasi oleh imajinasi manusia. Ingat restoran pizza dari bab terakhir - bagaimana mereka bisa memformat menu mereka? Ini bisa menjadi teks saja, daftar bullet; itu bisa menjadi serangkaian foto dengan keterangan; atau bahkan bisa hanya foto, yang pelanggan asing dapat menunjuk untuk menempatkan pesanan mereka.

Sebuah format yang dirancang dengan baik ditentukan oleh apa yang membuat informasi yang paling mudah untuk penonton dimaksudkan untuk memahami.

Prinsip yang sama berlaku ketika berbagi data antara komputer. Satu komputer harus menempatkan data dalam format yang lain akan mengerti. Umumnya, ini berarti semacam format teks. Format yang paling umum ditemukan di API modern JSON (JavaScript Object Notation) dan XML (Extensible Markup Language).

JSON

API baru telah mengadopsi JSON sebagai format karena itu dibangun pada bahasa pemrograman yang populer Javascript, yang di mana-mana di web dan dapat digunakan pada kedua depan dan back-end dari aplikasi web atau layanan. JSON merupakan format yang sangat sederhana yang memiliki dua buah: kunci dan nilai-nilai. Kunci mewakili atribut tentang objek yang sedang dijelaskan. Sebuah order pizza bisa menjadi objek. Ini memiliki atribut (kunci), seperti jenis kerak, topping, dan status pesanan. Atribut ini memiliki nilai yang sesuai (kerak tebal, pepperoni, dan keluar-untuk-pengiriman).

Mari kita lihat bagaimana agar pizza ini bisa terlihat di JSON:

```
{
    "crust": "original",
    "toppings": ["cheese", "pepperoni", "garlic"],
    "status": "cooking"
}
```

Dalam contoh di atas JSON, tombol adalah kata-kata di sebelah kiri: topping, kerak, dan status. Mereka memberi tahu kami apa atribut urutan pizza mengandung. Nilai-nilai adalah bagian sebelah kanan. Ini adalah rincian sebenarnya dari pesanan.

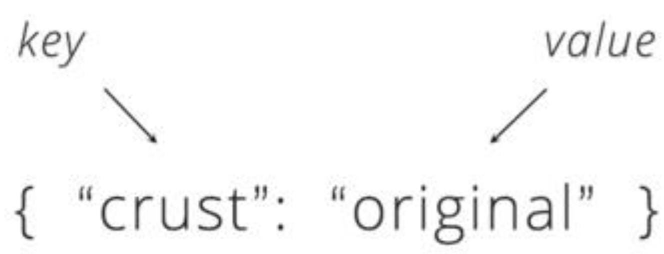


Figure 1. JSON key and value.

Jika Anda membaca garis dari kiri ke kanan, Anda mendapatkan kalimat bahasa Inggris yang cukup alami. Mengambil baris pertama sebagai contoh, kita bisa membacanya sebagai, "kerak pizza ini adalah gaya asli." Baris kedua bisa juga dibaca - di JSON, nilai yang dimulai dan diakhiri dengan tanda kurung siku ([]) adalah daftar nilai. Jadi, kita membaca baris kedua dari urutan sebagai, "topping untuk pesanan ini adalah: keju, pepperoni, dan bawang putih."

Kadang-kadang, Anda ingin menggunakan objek sebagai nilai kunci. Mari memperpanjang order pizza kami dengan rincian pelanggan sehingga Anda dapat melihat apa ini mungkin terlihat seperti:

```
{
  "crust": "original",
  "toppings": ["cheese", "pepperoni", "garlic"],
  "status": "cooking",
  "customer": {
    "name": "Brian",
    "phone": "573-111-1111"
  }
}
```

Dalam versi ini diperbarui, kita melihat bahwa kunci baru, "pelanggan", ditambahkan. Nilai untuk kunci ini adalah satu set kunci dan nilai-nilai yang memberikan rincian tentang pelanggan yang menempatkan pesanan. Keren trik, ya ?! Ini disebut Array asosiatif. Jangan biarkan istilah mengintimidasi teknis Anda meskipun - array asosiatif hanya benda bersarang.

XML

XML telah ada sejak tahun 1996 1. Dengan usia, itu telah menjadi format data yang sangat matang dan kuat. Seperti JSON, XML menyediakan blok bangunan sederhana yang pembuat API menggunakan struktur data mereka. Blok utama yang disebut node.

Mari kita lihat apa order pizza kami akan terlihat seperti di XML:

```xml
<order>
    <crust>original</crust>
    <toppings>
        <topping>cheese</topping>
        <topping>pepperoni</topping>
        <topping>garlic</topping>
    </toppings>
    <status>cooking</status>
</order>
```

XML selalu dimulai dengan simpul akar, yang dalam contoh pizza kami adalah "order." Dalam urutan yang lebih "anak" node. Nama masing-masing simpul memberitahu kita atribut pesanan (seperti kunci dalam JSON) dan data yang di dalam adalah detail yang sebenarnya (seperti nilai dalam JSON).

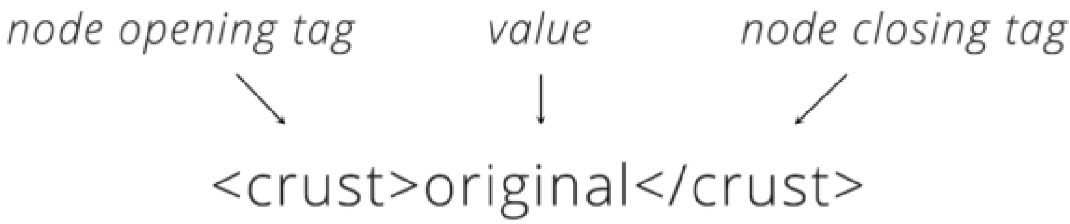


Figure 2. XML node and value.

Anda juga dapat menyimpulkan kalimat bahasa Inggris dengan membaca XML. Melihat sejalan dengan "kerak", kita bisa membaca, "kerak untuk pizza gaya asli." Perhatikan bagaimana dalam XML, setiap item dalam daftar topping dibungkus oleh sebuah node. Anda dapat melihat bagaimana format XML membutuhkan lebih banyak teks untuk berkomunikasi dari JSON tidak

Bagaimana data Format Apakah Digunakan Dalam HTTP

Sekarang kita sudah menjelajahi beberapa format data yang tersedia, kita perlu tahu bagaimana menggunakannya dalam HTTP. Untuk melakukannya, kita akan menyapa lagi ke salah satu dasar-dasar HTTP: header. Dalam Bab 2, kita belajar bahwa header adalah daftar informasi tentang permintaan atau respon. Ada sebuah header untuk mengatakan apa format data dalam: Content-Type.

Ketika klien mengirimkan Content-Type header dalam permintaan, itu memberitahu server bahwa data dalam tubuh permintaan diformat dengan cara tertentu. Jika klien ingin mengirim data server JSON, itu akan mengatur Content-Type untuk "aplikasi / json." Setelah menerima

permintaan tersebut dan melihat bahwa Content-Type, server pertama akan memeriksa apakah ia mengerti format yang, dan, jika demikian, itu akan tahu bagaimana membaca data. Demikian juga, ketika server akan mengirimkan klien respon, itu juga akan mengatur Content-Type untuk memberitahu klien cara membaca tubuh respon.

Kadang-kadang, klien hanya bisa berbicara satu format data. Jika server mengirimkan kembali apa pun selain format yang, klien akan gagal dan melemparkan kesalahan. Untungnya, header HTTP kedua datang untuk menyelamatkan. Klien dapat mengatur Terima header memberitahu server apa format data itu dapat menerima. Jika klien hanya bisa berbicara JSON, dapat mengatur Terima header "aplikasi / json." Server kemudian akan mengirim kembali respon dalam JSON. Jika server tidak mendukung format permintaan klien, dapat mengirim kembali kesalahan ke klien untuk membiarkannya tahu permintaan tidak akan bekerja.

Dengan dua header, Content-Type dan Terima, klien dan server dapat bekerja dengan format data mereka memahami dan perlu untuk bekerja dengan baik.

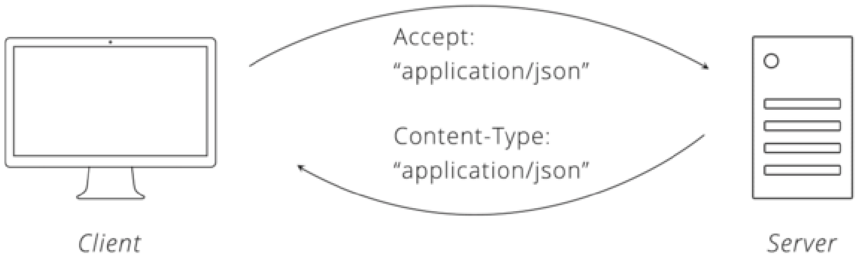


Figure 3. *Data format headers.*

Kesimpulan BAB 3

Dalam bab ini, kita belajar bahwa untuk dua komputer untuk berkomunikasi, mereka harus dapat memahami format data yang diberikan kepada mereka. Kami diperkenalkan ke 2 format data yang umum digunakan oleh API, JSON dan XML. Kami juga belajar bahwa Content-Type header HTTP adalah cara yang berguna untuk menentukan apa format data sedang dikirim dalam permintaan dan Terima sundulan menentukan format yang diminta tanggapan.

istilah kunci yang kita pelajari adalah:

- *JSON: JavaScript Object Notation*
- *Objek: hal atau benda (orang, order pizza ...)*
- *Key: atribut tentang suatu objek (warna, topping ...)*
- *Nilai: nilai atribut (biru, pepperoni ...)*
- *Array asosiatif: benda bersarang*
- *XML: Extensible Markup Language*

Bab 4(chafter) : Authentication, Bagian 1

Hal mulai mengambil dalam pemahaman kita tentang API. Kami tahu siapa klien dan server, kita tahu mereka menggunakan HTTP untuk berbicara satu sama lain, dan kita tahu mereka berbicara dalam format data yang spesifik untuk memahami satu sama lain. Mengetahui bagaimana berbicara, meskipun, meninggalkan pertanyaan penting: bagaimana server tahu klien yang klaim untuk menjadi? Dalam bab ini, kita mengeksplorasi dua cara bahwa klien dapat membuktikan identitasnya ke server.

Identitas dalam Dunia Virtual

Anda mungkin pernah terdaftar untuk account di situs web sebelumnya. Proses ini melibatkan situs yang meminta Anda untuk beberapa informasi pribadi, terutama username dan password. Kedua potongan informasi menjadi tanda identifikasi Anda. Kami menyebutnya kredensial Anda. Ketika Anda mengunjungi situs web lagi, Anda bisa login dengan menyediakan surat-surat ini.

Logging-in dengan username dan password adalah salah satu contoh dari proses teknis dikenal sebagai otentikasi. Ketika Anda mengotentikasi dengan server, Anda membuktikan identitas Anda ke server dengan mengatakan itu informasi yang hanya Anda yang tahu (setidaknya kami berharap hanya Anda tahu itu). Setelah server tahu siapa Anda, dapat mempercayai Anda dan membocorkan data pribadi di account Anda.

Ada beberapa teknik API gunakan untuk otentikasi klien. Ini disebut skema otentikasi. Mari kita lihat dua skema ini sekarang.

Otentikasi Dasar

Contoh logging-in di atas adalah bentuk paling dasar dari otentikasi. Bahkan, nama resmi untuk itu adalah Basic Authentication ("Basic Auth" untuk teman-temannya). Meskipun nama belum mengumpulkan penghargaan kreativitas, skema adalah cara yang bisa diterima untuk server untuk otentikasi klien dalam API.

Dasar Tupoksi hanya membutuhkan username dan password. Klien mengambil dua mandat ini, smooshes mereka bersama-sama untuk membentuk nilai tunggal 1, dan melewati itu bersama dalam permintaan dalam header HTTP yang disebut Kuasa

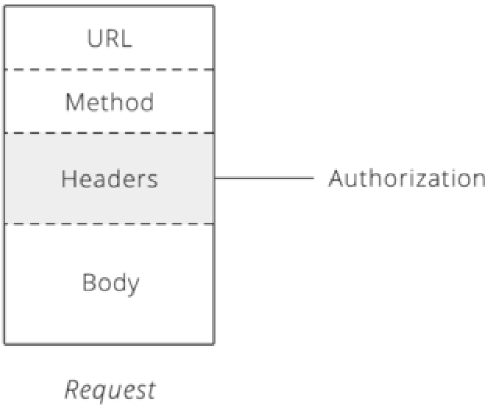


Figure 1. The Authorization HTTP header.

Ketika server menerima permintaan, terlihat di header Otorisasi dan membandingkannya dengan kepercayaan itu telah disimpan. Jika username dan password pertandingan salah satu pengguna dalam daftar server, server memenuhi permintaan klien sebagai pengguna tersebut. Jika tidak ada pertandingan, server mengembalikan kode status khusus (401) untuk membiarkan klien tahu bahwa otentikasi gagal dan permintaan tersebut ditolak.

Meskipun Basic Auth adalah skema otentikasi valid, fakta bahwa ia menggunakan username dan password yang sama untuk mengakses API dan mengelola account tidak ideal. Itu seperti sebuah hotel menyerahkan tamu kunci untuk seluruh bangunan daripada sebuah ruangan.

Demikian pula dengan API, mungkin ada saat-saat ketika klien harus memiliki hak akses yang berbeda dari pemilik akun. Ambil contoh seorang pemilik bisnis yang menyewa kontraktor untuk menulis sebuah program yang menggunakan API atas nama mereka. Mempercayai kontraktor dengan kredensial akun menempatkan pemilik beresiko karena kontraktor tidak bermoral bisa mengubah password, mengunci pemilik bisnis dari rekening mereka sendiri. Jelas, itu akan menyenangkan untuk memiliki alternatif.

API Key Authentication

API otentikasi kunci adalah teknik yang mengatasi kelemahan menggunakan kredensial bersama dengan mengharuskan API yang akan diakses dengan kunci yang unik. Dalam skema ini, kuncinya adalah biasanya serangkaian panjang huruf dan angka yang berbeda dari password login pemilik akun. Pemilik memberikan kunci kepada klien, sangat banyak seperti hotel memberikan tamu kunci untuk satu kamar.

Ketika klien mengotentikasi dengan kunci API, server tahu untuk memungkinkan akses klien ke data, tapi sekarang memiliki opsi untuk membatasi fungsi administratif, seperti mengubah password atau menghapus account. Kadang-kadang, kunci digunakan hanya sehingga pengguna tidak harus memberikan password mereka. Fleksibilitas yang ada dengan otentikasi API Key untuk membatasi kontrol serta password pengguna melindungi.

Jadi, mana kunci API pergi? Ada sebuah header untuk itu, juga, kan? Sayangnya, tidak ada. Tidak seperti Basic Auth, yang merupakan standar yang ditetapkan dengan aturan ketat, kunci API yang dikandung di beberapa perusahaan di hari-hari awal web. Akibatnya, otentikasi kunci API adalah sedikit seperti barat liar; setiap orang memiliki cara mereka sendiri melakukannya.

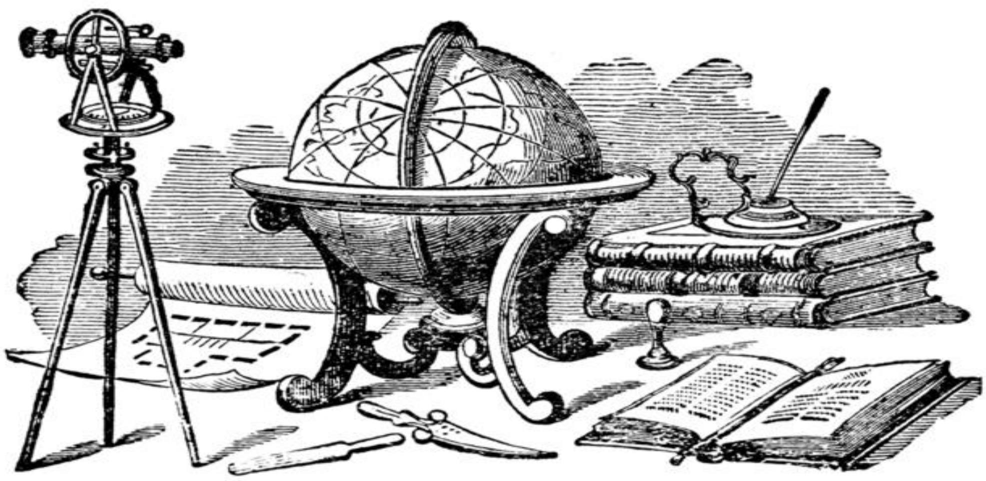
Seiring waktu, bagaimanapun, pendekatan umum beberapa telah muncul. Salah satunya adalah untuk memiliki klien memasukkan kunci di header Otorisasi, sebagai pengganti username dan password. Lain adalah untuk menambahkan kunci ke URL *(http://example.com?api_key=my_secret_key).* Kurang umum adalah untuk mengubur kunci di suatu tempat di tubuh permintaan sebelah data. Dimanapun kunci pepatah, efeknya adalah sama - itu memungkinkan server mengotentikasi klien.

Bab 4 Rekap

Dalam bab ini, kita belajar bagaimana klien dapat membuktikan identitasnya ke server, proses yang dikenal sebagai otentikasi. Kami melihat dua teknik, atau skema, API digunakan untuk otentikasi.

Istilah kunci yang kita pelajari adalah:

- Otentikasi: Proses klien membuktikan identitasnya ke server
- Kredensial: potongan rahasia informasi yang digunakan untuk membuktikan identitas klien (username, password ...)
- Dasar Auth: skema yang menggunakan nama pengguna dan password dikodekan untuk kredensial
- API Key Auth: skema yang menggunakan kunci yang unik untuk kredensial
- Otorisasi Header: header HTTP digunakan untuk menahan kredensial



Bab 5: Authentication, Bagian 2

Dalam Bab 4, kami menyebutkan paling website menggunakan username dan password untuk autentifikasi. Kami juga membahas bagaimana menggunakan kembali mandat ini untuk akses API tidak aman, sehingga API sering membutuhkan satu set yang berbeda dari kredensial dari yang digunakan untuk login ke website. Sebuah contoh umum adalah kunci API. Dalam bab ini, kita melihat solusi lain, Open Authorization (OAuth), yang menjadi skema otentikasi yang paling banyak digunakan di web.

Membuat Hidup Mudah untuk Orang

Pernahkah Anda harus mengisi formulir pendaftaran seperti di bawah?

Figure 1. *A product key as seen on Microsoft's Windows 8 registration form.*

Mengetik kunci panjang menjadi bentuk bidang seperti di atas membuat untuk miskin pengalaman pengguna. Pertama, Anda harus menemukan yang diperlukan kunci. Tentu, itu tepat di kotak masuk Anda ketika Anda membeli perangkat lunak, tetapi setahun kemudian, Anda berjuang untuk menemukan itu (Apa yang email itu dikirim dari? Email yang tidak saya gunakan untuk mendaftar ?!) Setelah berada, Anda harus masukkan hal terkutuk sempurna - membuat salah ketik atau hilang satu karakter akan menghasilkan kegagalan, atau bahkan mungkin membuat Anda terkunci keluar dari software terdaftar Anda!

Memaksa pengguna untuk bekerja dengan kunci API adalah pengalaman sama miskin. Typos adalah masalah umum dan mengharuskan pengguna untuk melakukan bagian dari setup antara klien dan server secara manual. Pengguna harus mendapatkan kunci dari server, kemudian memberikannya kepada klien. Untuk alat dimaksudkan untuk mengotomatisasi pekerjaan, pasti ada solusi yang lebih baik

Masukkan OAuth. Mengotomatisasi pertukaran kunci adalah salah satu masalah utama OAuth memecahkan. Ini menyediakan cara standar untuk klien untuk mendapatkan kunci dari server dengan berjalan pengguna melalui serangkaian langkah-langkah sederhana. Dari perspektif pengguna, semua OAuth membutuhkan memasuki kredensial. Di belakang layar, klien dan server yang berceloteh bolak-balik untuk mendapatkan klien kunci yang valid.

Saat ini ada dua versi OAuth, aptly bernama OAuth 1 dan OAuth 2. Memahami langkah-langkah di setiap diperlukan untuk dapat berinteraksi dengan API yang menggunakannya untuk otentikasi. Karena mereka berbagi alur kerja umum, kita akan berjalan melalui langkah-langkah OAuth 2, kemudian menunjukkan cara-cara yang OAuth 1 berbeda.

OAuth 2

Untuk memulai, pertama kita perlu mengetahui tokoh karakter yang terlibat dalam pertukaran OAuth:

- Pengguna - Seseorang yang ingin menghubungkan dua situs yang mereka gunakan
- Klien - Situs web yang akan diberikan akses ke data pengguna
- Server - Situs web yang memiliki data pengguna

Berikutnya, kita perlu memberikan disclaimer cepat. Salah satu tujuan dari OAuth 2 adalah untuk memungkinkan perusahaan untuk menyesuaikan proses otentikasi dengan kebutuhan mereka. Karena sifat diperpanjang ini, API dapat memiliki langkah-langkah yang sedikit berbeda. Alur kerja yang ditunjukkan di bawah adalah salah satu yang umum ditemukan di antara aplikasi berbasis web. Aplikasi mobile dan desktop mungkin menggunakan sedikit variasi pada proses ini.

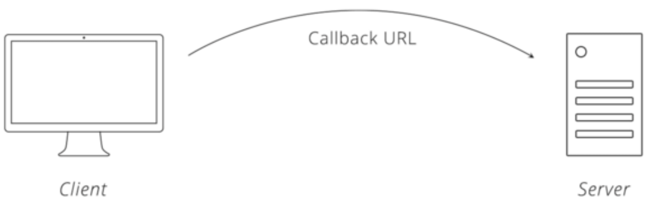
Dengan itu, berikut adalah langkah-langkah dari OAuth 2.

Langkah 1 - Pengguna Menceritakan Client Hubungkan ke Server

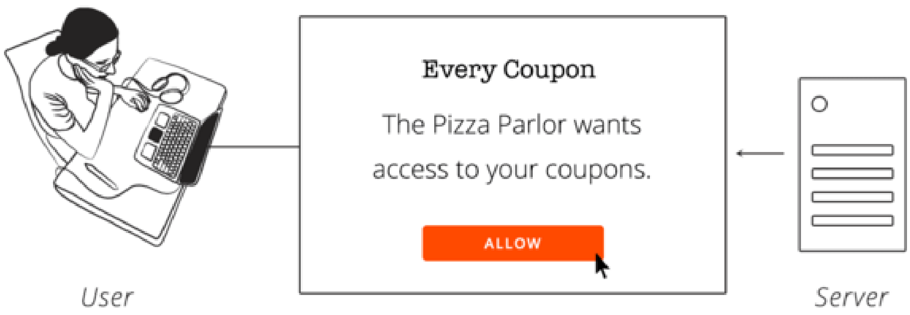
Pengguna kicks off proses dengan membiarkan klien tahu mereka ingin terhubung ke server. Biasanya, ini adalah dengan mengklik tombol.

Langkah 2 - Klien Mengarahkan Pengguna ke Server

Client mengirimkan pengguna ke situs web server, bersama dengan URL bahwa server akan mengirim pengguna kembali untuk sekali pengguna mengotentikasi, disebut URL callback.

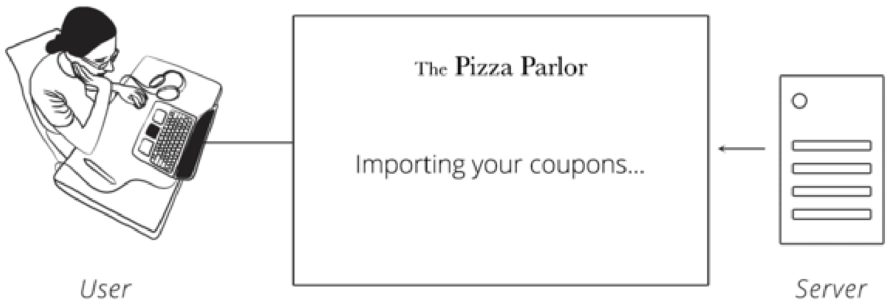


Langkah 3 - Pengguna Log-in ke Server dan Hibah Client Access

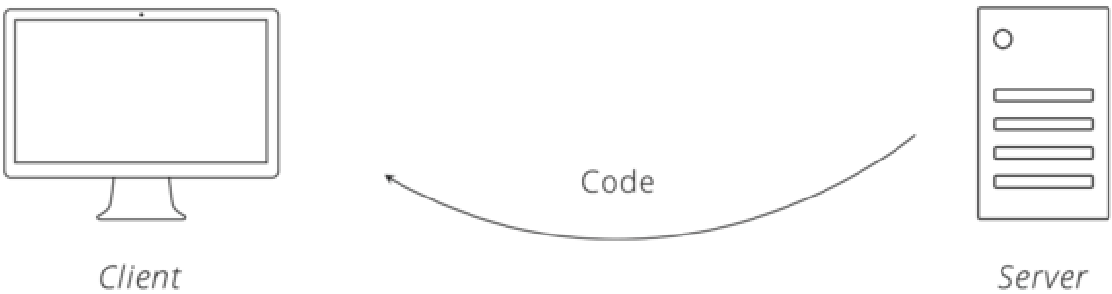


Dengan username dan password yang normal, pengguna mengotentikasi dengan server. Server sekarang yakin bahwa salah satu pengguna sendiri meminta bahwa klien akan diberikan akses ke akun pengguna dan data yang terkait.

Langkah 4 - Server Mengirim Pengguna Kembali ke Client, Seiring dengan Kode

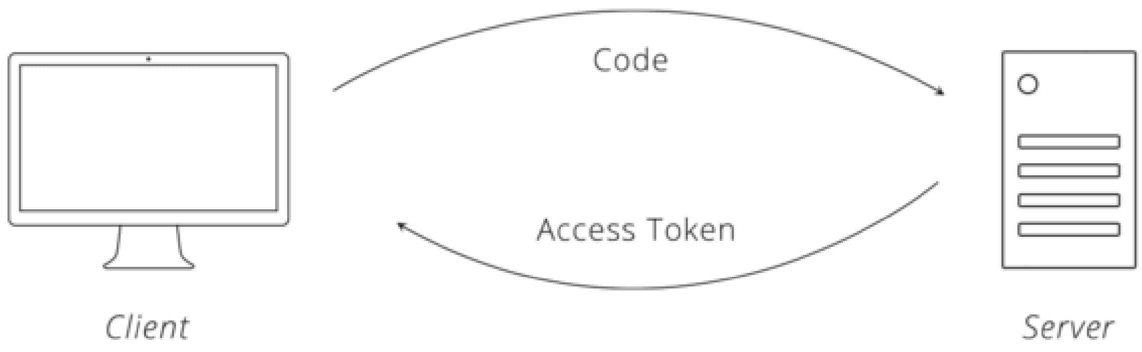


Server akan mengirimkan pengguna kembali ke klien (ke URL Callback dari Langkah 2). Tersembunyi di respon adalah kode otorisasi yang unik untuk klien.



Langkah 5 - Bursa Client Kode + Kunci Rahasia untuk Akses Token

Klien mengambil kode otorisasi yang diterimanya dan membuat permintaan lain ke server. Permintaan ini meliputi klien kunci rahasia. Ketika server melihat kode otorisasi yang valid dan klien dipercaya kunci rahasia, dapat dipastikan bahwa klien adalah yang mengklaim menjadi dan bahwa itu bertindak atas nama pengguna nyata. Server merespon kembali dengan token akses.



Langkah 6 - Klien mengambil data dari Server

Pada titik ini, klien bebas untuk mengakses server atas nama pengguna. Akses tanda dari Langkah 6 dasarnya sandi lain ke akun pengguna di server. Klien termasuk token akses dengan setiap permintaan sehingga dapat mengotentikasi langsung dengan server.

Klien Menyegarkan Token (pilihan)

Sebuah fitur diperkenalkan pada OAuth 2 adalah pilihan untuk memiliki token akses berakhir. Hal ini membantu dalam melindungi akun pengguna dengan memperkuat keamanan - lebih cepat token berakhir, kurang waktu token dicuri dapat digunakan jahat, mirip dengan bagaimana nomor kartu kredit berakhir setelah waktu tertentu. Umur dari token diatur oleh server. API di alam liar

menggunakan apa-apa dari jam ke bulan. Setelah umur tersebut tercapai, klien harus meminta server untuk token baru.

Bagaimana OAuth 1 Berbeda

Ada beberapa perbedaan utama antara versi OAuth. Salah satu yang kita telah disebutkan; token akses tidak berakhir.

Perbedaan lain adalah bahwa OAuth 1 meliputi langkah tambahan. Antara Langkah 1 dan 2 di atas, OAuth 1 memerlukan klien untuk meminta server untuk token permintaan. Token ini bertindak seperti kode otorisasi di OAuth 2 dan apa yang akan ditukar token akses.

Perbedaan ketiga adalah bahwa OAuth 1 membutuhkan permintaan untuk ditandatangani secara digital. Kami akan melewatkan rincian tentang bagaimana penandatanganan bekerja (Anda dapat menemukan perpustakaan kode untuk melakukan ini untuk Anda), tetapi itu sangat berharga mengetahui mengapa dalam satu versi dan bukan yang lain. Permintaan penandatanganan adalah cara untuk melindungi data dari yang dirusak sementara itu bergerak antara klien dan server. Tanda tangan memungkinkan server untuk memverifikasi keaslian permintaan.

Hari ini, Namun, sebagian besar lalu lintas API terjadi melalui saluran yang sudah aman (HTTPS). Menyadari hal ini, OAuth 2 menghilangkan tanda tangan dalam upaya untuk membuat versi dua lebih mudah digunakan. Trade-off adalah bahwa OAuth 2 bergantung pada langkah-langkah lain untuk memberikan keamanan untuk data dalam transit.

Otorisasi

Unsur OAuth 2 yang layak mendapat perhatian khusus adalah konsep membatasi akses, dikenal secara resmi sebagai otorisasi. Kembali pada Langkah 2, ketika pengguna mengklik tombol untuk memungkinkan akses klien, dimakamkan di baik cetak adalah izin yang tepat klien meminta. Hak akses, yang disebut ruang lingkup, adalah fitur penting lain dari OAuth 2. Mereka menyediakan cara bagi klien untuk meminta akses terbatas ke data pengguna, sehingga membuat lebih mudah bagi pengguna untuk percaya klien.

Apa yang membuat lingkup kuat adalah bahwa pembatasan berbasis client. Tidak seperti Key API, di mana batas-batas ditempatkan pada tombol mempengaruhi setiap klien yang sama, ruang lingkup OAuth memungkinkan satu klien untuk memiliki izin X dan izin lain X dan Y. Itu berarti satu situs mungkin dapat melihat kontak Anda, sementara situs lain dapat melihat dan mengeditnya.

Bab 5 Rekap

Dalam bab ini, kita belajar aliran proses otentikasi OAuth. Kami membandingkan dua versi, menunjukkan perbedaan besar antara mereka.

Istilah kunci yang kita pelajari adalah:

- OAuth: skema otentikasi yang mengotomatisasi pertukaran kunci antara klien dan server.
- Akses Token: rahasia bahwa klien mendapatkan atas berhasil menyelesaikan proses OAuth.

- Cakupan: perizinan yang menentukan apa yang mengakses klien memiliki data pengguna.

Bab 6: Desain API

Bab ini menandai titik balik dalam petualangan kami dengan API. Kami selesai meliputi fundamental dan sekarang siap untuk melihat bagaimana konsep sebelumnya bergabung membentuk sebuah API. Dalam bab ini, kita membahas komponen API dengan merancang satu.

pengorganisasian data

National Geographic memperkirakan bahwa pada tahun 2011, Amerika bentak 80 miliar foto 1. Dengan begitu banyak foto, Anda dapat membayangkan pendekatan yang berbeda orang harus mengorganisir mereka pada komputer mereka. Sebagian orang memilih untuk membuang segala sesuatu ke dalam satu folder. Lainnya cermat mengatur gambar mereka ke dalam hirarki folder dengan tahun, bulan, dan acara.

Perusahaan memberikan pemikiran yang sama dengan organisasi ketika membangun API mereka. Seperti yang telah disebutkan dalam Bab 1, tujuan API adalah untuk memudahkan bagi komputer untuk bekerja dengan data perusahaan. Dengan kemudahan penggunaan dalam pikiran, satu perusahaan mungkin memutuskan untuk memiliki URL tunggal untuk semua data dan membuatnya dapat dicari (semacam seperti memiliki satu folder untuk semua foto Anda). Lain mungkin memutuskan untuk memberikan setiap potongan data URL sendiri, yang diselenggarakan dalam suatu hirarki (seperti memiliki folder dan sub-folder untuk foto). Setiap perusahaan memilih cara terbaik untuk struktur API untuk situasi tertentu yang, dipandu oleh industri praktek terbaik yang ada.

Mulai dengan Gaya Arsitektur

Ketika mendiskusikan API, Anda mungkin mendengar pembicaraan tentang "sabun" dan "istirahat" dan bertanya-tanya apakah pengembang perangkat lunak melakukan pekerjaan atau merencanakan liburan. Yang benar adalah bahwa ini adalah nama-nama dari dua arsitektur yang paling umum untuk API berbasis web. SOAP (sebelumnya akronim 2) adalah desain berbasis XML yang telah dibakukan struktur untuk permintaan dan tanggapan. REST, yang merupakan singkatan dari transfer Representasi Negara, adalah pendekatan yang lebih terbuka, menyediakan banyak konvensi, tetapi meninggalkan banyak keputusan kepada orang merancang API.

Sepanjang kursus ini, Anda mungkin telah memperhatikan kami sudah kecenderungan untuk SISA API. Preferensi ini sebagian besar disebabkan tingkat yang luar biasa SISA ini adopsi 3. Ini bukan untuk mengatakan bahwa SOAP jahat; memiliki poin yang kuat 4. Namun, fokus diskusi kita akan tinggal di ISTIRAHAT karena ini kemungkinan akan menjadi semacam API Anda temui. Pada bagian yang tersisa, kami berjalan melalui komponen yang membentuk sebuah REST API.

Sumber Pertama kami

Kembali pada Bab 2, kita berbicara sedikit tentang sumber. Ingatlah bahwa sumber adalah kata benda dari API (pelanggan dan pizza). Ini adalah hal yang kita ingin dunia untuk dapat berinteraksi dengan melalui API kami.

Untuk mendapatkan merasakan bagaimana sebuah perusahaan akan merancang API, mari kita coba tangan kami di dengan ruang tamu pizza kami. Kita akan mulai dengan menambahkan kemampuan untuk memesan pizza.

Untuk klien untuk dapat berbicara pizza dengan kami, kita perlu melakukan beberapa hal:

- Memutuskan apa yang sumber daya (s) harus tersedia.
- Menetapkan URL untuk sumber daya.
- Memutuskan tindakan apa yang klien harus diizinkan untuk tampil di sumber daya.
- Mencari tahu apa potongan data yang diperlukan untuk setiap tindakan dan format apa yang mereka harus dalam.

Memilih sumber dapat menjadi sulit pertama tugas. Salah satu cara untuk mendekati masalah ini adalah untuk langkah melalui apa interaksi yang khas melibatkan. Untuk restoran pizza kami, kami mungkin memiliki menu. Pada menu yang pizza. Ketika pelanggan ingin kita untuk membuat salah satu pizza untuk mereka, mereka melakukan pemesanan. Dalam konteks ini, menu, pizza, pelanggan, dan ketertiban semua suara seperti kandidat yang baik untuk sumber daya. Mari kita mulai dengan pesanan.

Langkah selanjutnya adalah menetapkan URL ke sumber daya. Ada banyak kemungkinan, tapi untungnya konvensi SISA memberikan beberapa petunjuk. Dalam REST API yang khas, sumber daya akan memiliki dua pola URL yang ditugaskan untuk itu. Yang pertama adalah bentuk jamak dari nama sumber daya, seperti / perintah. Yang kedua adalah bentuk jamak dari nama sumber daya ditambah pengenal unik untuk menentukan sumber daya tunggal, seperti / perintah / <order_id>, di mana <order_id> adalah pengenal unik untuk memesan. Kedua pola URL membentuk endpoint pertama yang API kami akan mendukung. Ini disebut titik akhir hanya karena mereka pergi di akhir URL, seperti dalam `http://example.com/<endpoint_goes_here>`.

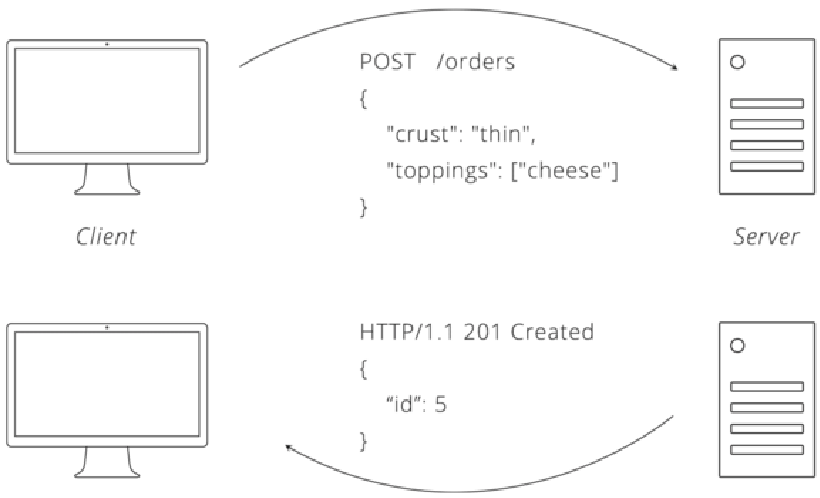
Sekarang kita mengambil sumber daya kami dan ditugaskan URL, kita perlu memutuskan tindakan apa yang klien dapat melakukan. Setelah konvensi REST, kita katakan bahwa endpoint jamak (/ order) untuk daftar perintah yang ada dan membuat yang baru. Jamak dengan identifier unik endpoint (/ order / <order_id>), adalah untuk mengambil, memperbarui, atau membatalkan urutan tertentu. Klien memberitahu server yang bertindak untuk melakukan dengan melewati kata kerja HTTP yang sesuai (GET, POST, PUT atau DELETE) dalam permintaan.

Secara keseluruhan, API kami sekarang terlihat seperti ini:

HTTP verb	Endpoint	Action
GET	/orders	List existing orders
POST	/orders	Place a new order
GET	/orders/1	Get details for order #1
GET	/orders/2	Get details for order #2
PUT	/orders/1	Update order #1
DELETE	/orders/1	Cancel order #1

Dengan tindakan untuk endpoint pesanan kami fleshed keluar, langkah terakhir adalah untuk memutuskan apa data perlu dipertukarkan antara klien dan server. Meminjam dari contoh pizza kami dalam Bab 3, kita dapat mengatakan bahwa perintah membutuhkan kerak dan topping. Kita juga perlu memilih format data bahwa klien dan server dapat digunakan untuk menyampaikan informasi ini bolak-balik. XML dan JSON keduanya pilihan yang baik, tapi demi dibaca, kita akan pergi dengan JSON.

Pada titik ini, Anda harus menepuk diri di bagian belakang; kami telah merancang sebuah API fungsional! Berikut adalah apa interaksi antara klien dan server mungkin terlihat seperti menggunakan API ini:



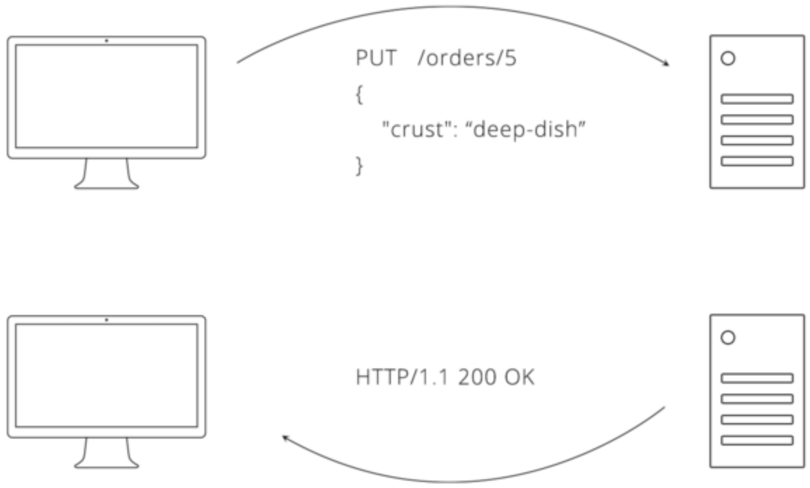


***Figure 1.** Example interaction between the client and server using our API.*

Menghubungkan Sumber Daya Bersama

Kami pizza API tampak tajam. Pesanan datang di tidak seperti sebelumnya. Bisnis ini sangat baik sebenarnya, kami memutuskan kami ingin memulai pelacakan pesanan oleh pelanggan untuk mengukur loyalitas. Cara mudah untuk melakukan ini adalah untuk menambahkan sumber daya pelanggan baru.

Sama seperti dengan perintah, sumber daya pelanggan kami membutuhkan beberapa titik akhir. Setelah konvensi, `/customers` and `/customers/<id>` cocok dengan baik. Kami akan melewatkan rincian, tetapi katakanlah kita memutuskan tindakan yang masuk akal untuk setiap endpoint dan data apa mewakili pelanggan. Dengan asumsi kita melakukan semua itu, kami datang ke pertanyaan menarik: bagaimana kita kaitkan pesanan dengan pelanggan?

Praktisi SISA dibagi tentang bagaimana untuk memecahkan masalah menghubungkan sumber daya. Beberapa mengatakan bahwa hirarki harus terus tumbuh, memberikan titik akhir `/customers/5/orders` pesanan untuk semua pelanggan # 5 ini perintah `/customers/5/orders/3` untuk pelanggan # 5 ini urutan ketiga. Yang lain berpendapat untuk menjaga hal-hal datar dengan memasukkan rincian terkait dalam data untuk sumber daya. Dalam paradigma ini, menciptakan perintah memerlukan lapangan `customer_id` untuk dikirim dengan rincian pesanan. Kedua solusi digunakan oleh SISA API di alam liar, sehingga layak mengetahui tentang masing-masing.

```
POST /customers/5/orders/3
{
  "crust": "thin",
  "toppings": ["cheese"]
}
```

VS

```
POST /orders/3
{
  "crust": "thin",
  "toppings": ["cheese"],
  "customer_id": 5
}

GET /customers/5
{
  "name": "Brian"
}
```

Figure 2. *Two ways to handle associated data in API design.*

Mencari data

Sebagai data dalam suatu sistem tumbuh, endpoint yang daftar semua catatan menjadi tidak praktis. Bayangkan jika restoran pizza kami memiliki tiga juta pesanan selesai dan Anda ingin mengetahui berapa banyak memiliki pepperoni sebagai topping. Mengirimkan permintaan GET ke / perintah dan menerima semua tiga juta pesanan tidak akan sangat membantu. Untungnya, SISA memiliki cara yang bagus untuk mencari melalui data.

URL memiliki komponen lain yang kami belum disebutkan belum, string query. Permintaan berarti pencarian dan string yang berarti teks. Query string adalah sedikit teks yang masuk ke akhir URL untuk lulus hal-hal bersama ke API. Misalnya, semuanya setelah tanda tanya adalah string di

`http://example.com/orders?key=value`.

API SISA menggunakan string query untuk menentukan rincian pencarian. Rincian ini disebut parameter permintaan. API menentukan apa parameter akan menerima, dan nama-nama yang tepat dari parameter perlu digunakan bagi mereka untuk mempengaruhi pencarian. Kami pizza API dapat memungkinkan klien untuk mencari pesanan dengan topping dengan menggunakan URL ini:

`http://example.com/orders?topping=pepperoni` Klien dapat mencakup beberapa parameter permintaan dengan daftar satu demi satu, memisahkan mereka dengan ampersand ("&"). Sebagai contoh `http://example.com/orders?topping=pepperoni&crust=thin`.

Penggunaan lain dari query string adalah untuk membatasi jumlah data yang dikembalikan dalam setiap permintaan. Seringkali, API akan membagi hasil ke set (mengatakan dari 100 atau 500

catatan) dan kembali satu set pada suatu waktu. Proses pemisahan up data ini dikenal sebagai pagination (analogi untuk putus kata dalam halaman buku). Untuk memungkinkan klien untuk halaman melalui semua data, API akan mendukung parameter permintaan yang memungkinkan klien untuk menentukan halaman data yang diinginkan. Dalam kami restoran pizza API, kita dapat mendukung paging dengan memungkinkan klien untuk menentukan dua parameter: halaman dan ukuran. Jika klien membuat permintaan seperti GET / perintah? Page = 2 & size = 200, kita tahu mereka ingin halaman kedua dari hasil, dengan 200 hasil per halaman, sehingga perintah 201-400.

Bab 6 Rekap

Dalam bab ini, kita belajar bagaimana merancang sebuah REST API. Kami menunjukkan fungsi dasar API mendukung dan bagaimana untuk mengatur data sehingga dapat dengan mudah dikonsumsi oleh komputer.

Istilah kunci yang kita pelajari adalah:

- SOAP : Arsitektur API dikenal untuk format pesan standard
- ISTIRAHAT: arsitektur API yang berpusat di sekitar sumber memanipulasi
- Sumber daya : Istilah API untuk kata benda bisnis seperti pelanggan atau perintah
- Endpoint: Sebuah URL yang membentuk bagian dari API. Dalam REST, setiap sumber daya mendapat endpoint sendiri
- Query String: Sebagian dari URL yang digunakan untuk melewatkan data ke server
- Parameter Permintaan: Sepasang kunci-nilai yang ditemukan dalam string (topping keju =)
- Pagination: Proses memecah hasil dalam potongan dikelola

PENUTUP

Ebook ini adalah terjemahan dari webisite https://zapier.com/learn/apis/ dan saya coba translate kedalam bahasa Indonesia, semoga bermanfaat.

Website : www.belajarphp.net

Belajarphp.net – cara mudah belajar coding

dapatkan suasana kursus web programming yang harganya jutaan rupiah sekarang jadi lebih terjangkau sekarang juga, mereka sudah membuktikan bahwa belajar web development itu mudah, sekarang giliran anda , dapatkan disini http://store.belajarphp.net/toko/

TESTIMONI PELANGGAN